RAPPORT OPTIMISATION

Détection de biomes sur des exoplanètes

1. Travail effectué

a. Flou gaussien

Toutes les cartes ont été traitées selon un algorithme de flou gaussien avant d'y appliquer les 3 algorithmes suivants.
Ceci a permis de retirer le bruit de l'image, tous les pixels parasites, les défauts, tout ce qui peut provoquer des incohérences avec les algorithmes suivants a été lissé.
L'application du flou gaussien sur une image avant son traitement supprime les détails inutiles et permet aux algorithmes d'être plus efficaces, ils concentrent alors leur traitement sur les structures importantes et non sur les détails qui pourraient donner de faux résultats et rendre les algorithmes moins performants.

b. Kmeans

Pour la majorité des détections des biomes, nous avons utilisé l'algorithme KMeans en raison de sa simplicité et rapidité. Après avoir appliqué un flou gaussien pour réduire le bruit, nous avons segmenté les données en fixant un nombre de clusters prédéfini (souvent 5).

KMeans répartit les pixels en groupes homogènes selon leurs caractéristiques colorimétriques, ce qui permet d'identifier des biomes distincts. Ce choix s'est avéré pertinent car il offre un bon compromis entre performance et qualité de segmentation, malgré une séparation linéaire qui limite la complexité des formes détectées.

c. DBScan

Après la détection des biomes par KMeans, nous appliquons DBSCAN uniquement sur les coordonnées des pixels appartenant à un même biome. Deux paramètres clés guident l'algorithme :

- ε (eps) : distance maximale entre deux points pour former un cluster (fixée entre 5 et 15 pixels selon les cartes)
- minPts : nombre minimal de points pour constituer un cluster (+/- 10)

Nous avons optimisé epsilon a nécessité une sur la distance moyenne entre plus proches voisins, minPts a aussi été calibré pour éviter la détection de trop de micro-régions.

L'intérêt principal de DBSCAN est de détecter automatiquement le nombre de régions, d'ignorer les points isolés (le bruit restant), de capturer des formes complexes (lacs, forêts irrégulières)

Son inconvénient majeur est que son temps d'exécution augmente drastiquement selon la taille des biome.

Résultats
En combinaison avec KMeans pour les biomes, DBSCAN produit des régions

- Plus naturelles que la segmentation linéaire de KMeans seul
- Moins gourmandes en mémoire que HAC
- Adaptatives à la densité locale (détection fine des frontières)

d. HAC et EHAC

Dès le départ, nous avions déterminé l'enjeu qu'aller être d'implémenter ce projet dans les temps. C'est pourquoi, l'algorithme le plus rapide a implémenté (Kmeans) a été effectué par la personne qui allez mettre les fondations au projet (main, interface etc...), celle qui a effectué DBScan s'est occupé des fonctions intermédiaires nécessaires à chacun (flou par exemple), et une autre a été mise sur HAC dès le départ afin que tout le projet se termine dans les temps.

HAC est un algorithme qui nécessite plus d'ample compréhension de son fonctionnement pour être implémenté. Grâce à notre organisation, nous avons même eu le temps de proposer une amélioration à cet algorithme (voir section dédiée à HAC).

Cet algorithme, bien que très gourmand, nous a en tout point impressionné sur les résultats qu'il pouvait produire. Ce sont donc le défi de sa mise en œuvre et l'appétence de ses résultats qui ont poussé notre groupe a maintenir l'objectif de son implémentation et de son amélioration.

e. Biomes

Pour la définition des biomes, chaque cluster correspond à un biome spécifique, caractérisé par une homogénéité visuelle (une "palette de couleur"). Le choix du nombre de clusters a été calibré pour équilibrer le nombre des biomes et la lisibilité des résultats. Cela offre une base robuste pour les étapes suivantes de classification des régions.

f. Régions

Pour la classification des régions, nous avons comparé plusieurs méthodes, privilégiant DBScan et HAC selon les cas. La particularité de notre approche est que nous appliquons ces algorithmes **sur chaque biome détecté**.

Lors de cette seconde phase, au lieu d'utiliser directement les couleurs, nous exploitons les coordonnées spatiales des pixels appartenant à un même biome pour extraire des régions cohérentes à l'intérieur de ces zones. Cette méthode en deux temps permet ainsi de mieux refléter la structuration naturelle des régions, en affinant la segmentation après la détection initiale des biomes.

2. Analyses

1. Biomes KMeans + Régions KMeans

Performance : Plutôt rapide.
Efficacité de Séparation : La séparation entre les régions est peu efficace, souvent linéaire en raison des limitations de l'algorithme KMeans.
Conclusion : Convient pour des analyses rapides mais moins efficace pour des séparations complexes.

2. Biomes KMeans + Régions DBScan

Performance : Lent lors de l'analyse des régions.
Efficacité de Séparation : Séparation plutôt efficace et plus logique que KMeans seul.
Conclusion : Bon compromis entre performance et efficacité de séparation, bien que le temps de traitement soit un inconvénient.

3. Biomes KMeans + Régions HAC Average

Performance : Lent lors de l'analyse des régions.
Efficacité de Séparation : Très bonne séparation des régions.
Conclusion : Efficace pour des séparations précises, mais la lenteur peut être un facteur limitant. Sans négliger la taille de l’image qui peut devenir un vrai problème de calcul.

4. Biomes HAC Average + Régions HAC Average

Performance : Très lent.
Efficacité de Séparation : Très efficace, mais l'utilisation de HAC sur les biomes peut être considérée comme excessive ("overkill").
Conclusion : Bien que très efficace, cette méthode est peu pratique en raison de sa lenteur. KMeans pour les biomes offre un rendu similaire avec une meilleure performance.

5. Optimisation de HAC (modification des paramètres)

Pour optimiser l'algorithme de classification hiérarchique ascendante (HAC), nous avons mené une série d'expériences en faisant varier les paramètres un par un sur une même carte. L'objectif était de déterminer la configuration offrant les meilleures performances pour notre application spécifique.

Méthodologie

Nous avons testé différentes méthodes de liaison :

- **SINGLE** : Cette méthode utilise la distance minimale entre les clusters.
- **COMPLETE** : Cette méthode utilise la distance maximale entre les clusters.
- **AVERAGE** : Cette méthode utilise la distance moyenne entre les clusters.
- **CENTROID** : Cette méthode utilise la distance entre les centroïdes des clusters.

En plus des méthodes de liaison, nous avons également fait varier le nombre de clusters pour observer comment cela affectait la performance et la qualité de la segmentation.

Résultats et Observations

En comparant les résultats obtenus avec les différentes configurations, nous avons observé que les méthodes **AVERAGE et CENTROID** offraient souvent les meilleurs résultats. Cette méthode a montré un bon équilibre entre la qualité de la segmentation et la cohérence des résultats.

Réflexion sur la performance

En effet, nous avons remarqué que HAC était très gourmand en mémoire.
A l'aide de calcul (dont la plupart effectué en classe à l'aide du professeur) nous pouvons estimer que pour calculer une image de 400 par 400 pixels, il faudrait environ 250 Go de mémoire.
Ce constat effectué, une recherche de notre part a été menée pour essayer de rendre HAC plus léger et plus optimisé. Pour ce faire nous avons développé (à l'aide de forum aussi notamment) une version de HAC qui au lieu de stocker chaque pixels, va venir conservé une couleur dans un dictionnaire (Map en Java) et incrémente sa valeur à chaque répétition de cette couleur au lieu de restaurer entièrement cette valeur.
Les vidéos proposant cette optimisation estiment un gain passant de la centaine de Go de mémoire à la dizaine de Mo de mémoire. Une amélioration totalement nécessaire afin de finaliser

3. Annexes

KMEANS BIOMES (clusters: 5) + DBSCAN REGIONS

KMEANS BIOMES (clusters: 5) + HAC (Average, clusters: 5) REGIONS

KMEANS BIOMES (clusters: 5) + KMEANS REGIONS (clusters: 5)